When a compiler compiles a program into machine language or byte code, it usually performs optimizations to improve the speed or memory usage of the code. For example, the compiler may attempt to remove unnecessary code, that will never be called. The following problem places a theoretical limit on that kind of optimization. Define $ENTER_{TM}$ to be the set of all strings $\langle M, q\rangle$, where M is a Turing machine, q is a state of M, and there exists a string w such that M enters q while processing input w.

A. Prove that $ENTER_{TM}$ is undecidable.

This next problem is like Problem 5.31 but easier. To understand it, first read the description of the $3x + 1$ problem in Problem 5.31. (It's a notorious unsolved problem in mathematics. We talked about it on the first day of this course.) Then, fix the input alphabet $\Sigma = \{0, 1\}$, and let

$$ALL_{TM} = \{\langle M\rangle : M \text{ accepts all strings over } \Sigma\}.$$

B. Suppose that you had a decider H for ALL_{TM}. Explain how you could use H to design a decider D that outputs the answer to the $3x + 1$ problem.

Recall that, for any two languages A and B,

$$A/B = \{w : \exists x \in B \text{ such that } wx \in A\}.$$

In an earlier assignment, we showed that if A is regular then A/B is regular. That proof was non-constructive. This next problem shows that the proof *must* be non-constructive — because it must be non-constructive even in the relatively tame case where B is recognizable.

Imagine a Turing machine N. It is designed to take input $\langle D, M\rangle$, where D is a DFA and M is a Turing machine. Somehow N processes this input $\langle D, M\rangle$, eventually halting with a string $\langle C\rangle$ on its tape, and nothing else. Here, C is a DFA such that $L(C) = L(D)/L(M)$. To be clear: N halts on all inputs, but we do not care whether it accepts or rejects them. We care about the contents of the tape upon halting.

C. Prove that such an N cannot exist. (Hint: $EMPTY_{TM}$.)